A new hardware stack sequence decoder based on FPGA

Wenliang Zou[a*], Yuzhong Jiang[a], Xiaoyong Li[b]
[a] School of Electronic Engineering, Naval University of, Wuhan 430033, China; [b] Electrical Engineering, Naval University of Engineering, Wuhan 430033, China

ABSTRACT

Compared with the Viterbi algorithm, the stack algorithm can provide lower hardware complexity, especially for long constraint length convolutional codes. This paper proposes a fast and simple hardware stack sequence decoder with an efficient state scheme. The stack decoder structure is mainly composed of RAM and shift register, and three independent RAM parts store the path metric, node, and encoder state of each path. Accessing different data items of the same stack in the data structure can be achieved by addressing the RAM with the same register value. In the decoding process, the paths are sorted according to the rules of the stack algorithm, and the path located at the top of the stack will execute the state of path extension in the next clock cycle. In this paper, an FPGA prototype of the stack decoder is constructed, and high-speed decoding is obtained by optimizing the state scheme, avoiding additional time-consuming read/write operations.

Keywords: Long constraint length, stack algorithm, FPGA

1. INTRODUCTION

The Viterbi algorithm is widely cited as the optimal decoding method for convolutional codes, but its memory requirement is proportional to the constraint length of the convolutional codes[1]. Long constrained length convolutional codes can provide strong error correction ability and keep low error probability in the environment of extremely low signal-to-noise ratio, and are widely used in ultra-long distance communication. such as satellite communications[2-3]. In order to apply the Viterbi algorithm to convolutional codes with long constraint length, a large Viterbi decoder (BVD) built by the Jet Propulsion Laboratory in the United States in 1991 can decode one million bits in one second, making Convolutional codes with a constraint length of 15 are used in Mars Pathfinder, Saturn Cassini rover, etc. But to decode the convolutional codes with constraint length of 32 or even 64, sequence decoding is the best choice[4-6].

The complexity of the sequence decoding algorithm has nothing to do with the length of the constraint. In addition, the structure is simple, and the burden on the communication system is small. Therefore, decoding is performed using a sequence decoding algorithm whose decoding complexity and constraint length are independent[7-8]. Regarding the hardware implementation of sequence decoding, there are many FPGA implementations of Fano algorithm and its improved algorithm in the literature. However, the hardware implementation of the stack algorithm is relatively small, and the time is earlier, and the implementation on FPGA is even less[9-12]. The stack algorithm is a classical sequence decoding algorithm with a simple logical structure. Each node will only be checked once, and sufficient stack size will result in good decoder performance. It requires more memory than Fano algorithm, but it is far less than Viterbi algorithm[13].

This paper presents a stack decoder for (2, 1, 31) convolutional codes and soft-decision multilevel quantization. The frame length N (including the tail bits in the tail) can vary, but is usually set to 64 bits, including a fixed tail mode of 32 bits. The stack decoder is constructed by using shift register and RAM, and the corresponding state machine is designed. Because the Verilog language is particularly suitable for describing complex combinational logic, group operations and state machines, truth tables and parameterized logic, Verilog programming is used. After debugging and simulation, it is finally downloaded to the FPGA to run.

2. DESIGN OF STACK DECODER

2.1 Stack algorithm flow

As a kind of sequence decoding algorithm, the stack algorithm has a simpler logical structure than the Fano algorithm,

* 1668166203@qq.com; zwlhgpaper@163.com

Third International Conference on Computer Science and Communication Technology (ICCSCT 2022)
edited by Yingfa Lu, Changbo Cheng, Proc. of SPIE Vol. 12506, 125065W
 · 0277-786X · doi: 10.1117/12.2661847

but requires more memory. The memory is mainly used to store all likelihood paths in the decoding process. For the fixed tail of the (2,1,31) convolutional code used in this paper, the decoding process is shown in Figure 1. When decoding starts, the stack needs to be initialized, and then the decoding process can be roughly divided into three steps[14-15].

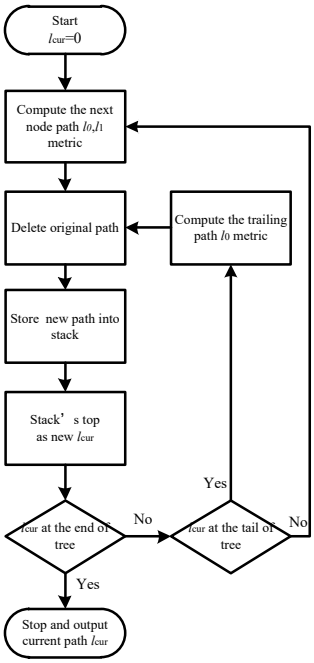

Figure 1. Flow chart of the stack algorithm.

(1) Extend the current path *lcur* and delete this path in the stack. In general, the next node symbol is 0 or 1, and the path after expansion is l_0, l_1. But when the node is at the end of the tree, only one possible path l0 needs to be calculated.

(2) The new path after expansion is stored in the stack according to the size of the path metric using insertion sort, and the path at the top of the stack is used as the new *lcur*.

(3) Judge the node of the current path *lcur* in the code tree. If the node is before the fixed tail, go back to step 1 and keep two extended paths l_0, l_1; if the node is between the tail and the end, go back to step 1 and keep only one extended path l0, if the node is the end of the tree, then End decoding, and output the current path *lcur*.

2.2 Structure of stack decoder

The stack decoder structure is mainly composed of RAM and shift register, and three independent RAM parts store the path metric, node, and encoder state of each path[16-17]. The shift register stores the height in the stack, and addresses the path metrics, nodes, and encoder states of the RAM through the same shift register value, so as to realize the access of different data items in the same stack in the data structure. According to the stack algorithm flow design, its structure is shown in Figure 2.

The Stack decoder control module is the brain of the decoder, and issues instructions to control the start, end and sequence of decoding based on feedback. In symbols are symbols to be decoded. After receiving the start command of the control module, they are input into the node metric module. In the encoder module, according to the encoder state and the input symbol 0 or 1 of the next node, the encoding symbol corresponding to the next node can be obtained. In the absence of other conditions, the next node symbol may be 0 or 1, so there are two possibilities for the encoding symbol, both of which are input into the node metric module for mapping. In the node metric module, the coded symbols and in symbols obtained from the encoder state are mapped in mettab to obtain the next node metric. The path metric is accumulated by each node metric, and the new path will be stored in the built stack. The shift register value sth represents the position in the stack. As the new path is generated, the encoder state is shifted to the left by one bit from the encoder state of the original path, the next node symbol is input, the node value is increased by one, and the two are

used as the stack. The different data items of the element are stored in the corresponding RAM according to the sth value following the path metric. According to the new path metric within the path metric, the input to the compute module is sorted. For sorted datasets, insertion sort is a very low time-complexity sorting method, so stack element sorting uses this method. The stack decoder control module changes the sth value according to the feedback of the compute module, so that the stack elements are exchanged, so as to achieve the purpose of sorting the stack elements. As the decoding progresses, the nodes located in different positions of the code tree have different effects, so the Stack decoder control module controls the number of new paths and the end of the stack according to the value stored in the node. Finally, when the node value at the top of the stack is at the end of the code tree, the Stack decoder control module sends an instruction to the encoder state storage part to output the state of the top of the stack to obtain the decoded symbol decdata.

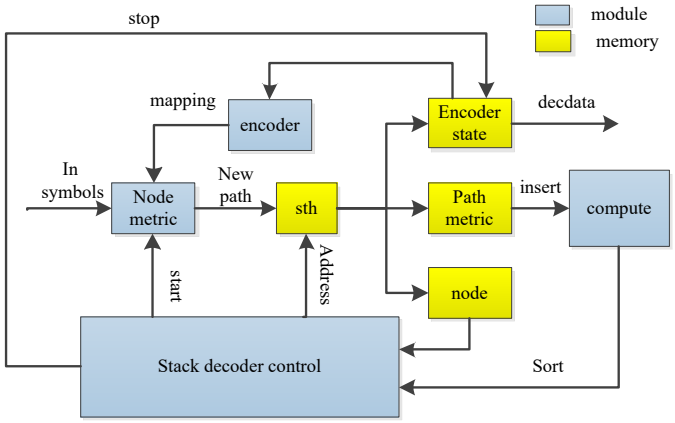

Figure 2. Main structure of stack decoder.

3. THE OPERATION OF HIGH SPEED ARCHITECTURE

Combined with the state flow chart and the decoder structure diagram, the flow chart shown in Figure 1 is classified and merged, and the finite-state machine (FSM) is designed to realize the function of the stack decoder as shown in Figure 3.

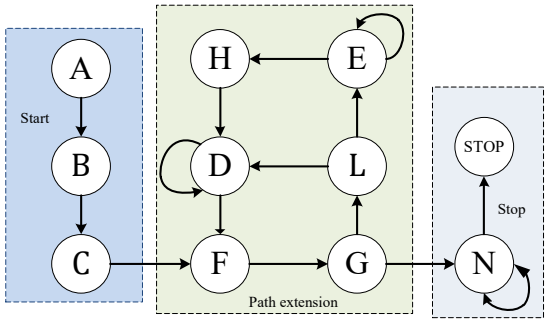

Figure 3. FSM describing the RTL.

The state machine can be divided into three stages, namely start, path extension and stop. The start stage has three states, A initializes the stack space, and the shift register value sth takes 0. B pairs the first node of the symbol tree. code element to explore. C is successively stored on the stack according to the metric size.

Then, enter the path extension stage. This is the body of the decoding process, finding the largest possible path to the end of the code tree. F takes out the top path of the stack as lcur, and explores the current path to the next node. G judges the position of the next node on the code tree. If it reaches the end point, it jumps to the stop stage for decoding, otherwise it enters the next state L. L compute the metric of the extended path l0, and judges the position of the next node on the code tree. If it reaches the tail of the tree, it jumps to state D, otherwise it jumps to state E. D performs insertion sorting on the

input expansion path and stores it in the stack, the path input by L is l0, and the path input by H is l1. E performs insertion sorting on the extended path l0, stores it in the stack, and jumps to the H state. H computes the metric of the extended path l1and inputs it to D for sorting.

After multiple path expansions, the current path lcur node reaches the end of the tree, and then jumps from state G to the Stop stage. N decodes and outputs the encoder state of lcur, outputs one symbol each time, enters the stop state after several cycles, and ends the decoding.

4. FPGA IMPLEMENTATION OF THE STACK SEQUENTIAL DECODER

In this study, we set the length of each frame of the (2, 1, 31) convolutional code to 64, including a 32-bit fixed tail, and the upper limit of the number of stacks to 500. We use co-simulation to complete the soft-decision stack decoder in FPGA work, testbench as shown in Figure 4. We use Dev-c++ to simulate the communication system and generate random codewords. The codeword is encoded by a convolutional encoder and modulated into BPSK symbols. The modulated data passes through the AWGN channel. Some noise is applied in this channel by taking into account the SNR value entered. The demodulated symbols enter the decoder for decoding, and the obtained decoded symbols and the original random codeword are input into the statistics module, and the bit error rate under the corresponding signal-to-noise ratio can be obtained. The demodulated signal and noise data are input to the test bench written in Verilog, after decoding by the written stack decoder, the waveforms of symbols, path metrics, nodes, encoder states and the state of the state machine are decoded Shown on Modelsim SE-64 10.7.

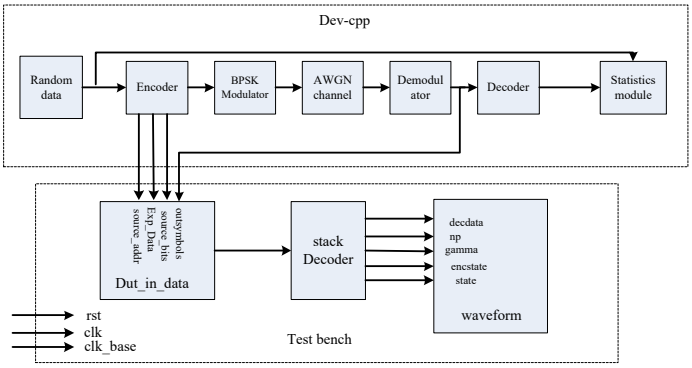

Figure 4. Simulation setup.

When the signal-to-noise ratio is 3, the single simulation result in the FPGA is shown in Figure 5. The soft decision stack decoder successfully completed the decoding, and only one frame error code appeared in 1000 frames of information [18]. After modifying the parameters such as the signal-to-noise ratio and testing, it is found that the decoding results are the same as the software side. The FPGA design of the soft-decision stack decoder is successful. Download the program to the ax7020 development board of xlinx company, and the decoding results observed by the logic analyzer of vivado software are shown in Figure 6.

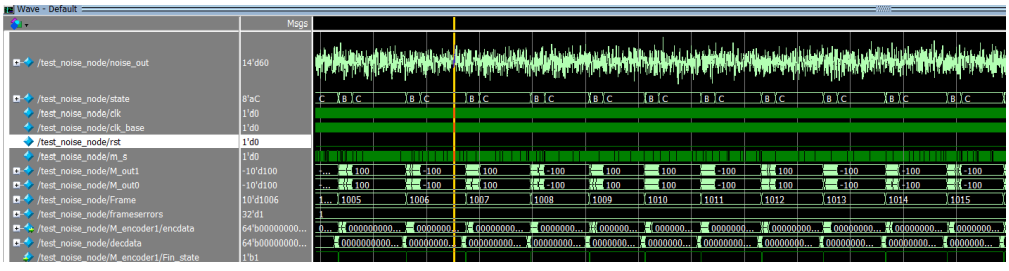

Figure 5. Simulation report of Modelsim-SE 10.7.

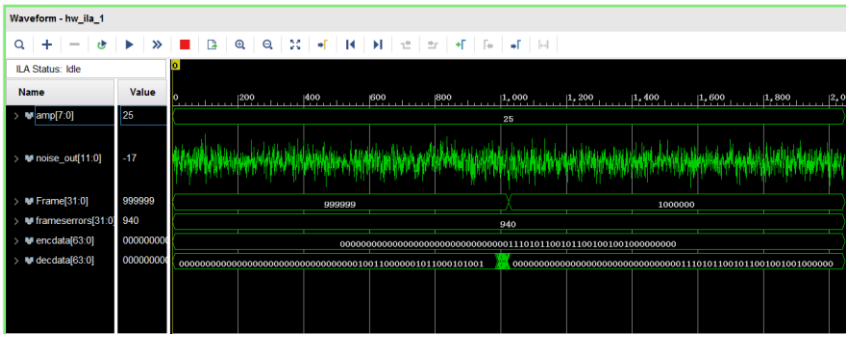

Figure 6. Decoding result of one million frames.

5. CONCLUSION

In this paper, the FPGA prototype of the stack decoder is constructed by using the shift register and RAM, and an effective state scheme is given. The architecture of the decoder has been developed into a Verilog core, which is further implemented on an FPGA device. High-speed decoding is achieved through an optimized state scheme, avoiding additional time-consuming read/write operations. The FPGA implementation of the Stack decoder provides a new direction for hardware design using convolutional code decoders. For the follow-up research, the multi-core performance of FPGA can be used to decode multiple stack expansion paths at the same time to improve the decoding efficiency.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation for Outstanding Young of China under Grant 42122025.

REFERENCES

[1] Yoshikawa, H., “Error performance analysis of the K-best Viterbi decoding algorithm,” 2018 International Symposium on Information Theory and Its Applications (ISITA), IEEE, 257-260(2018).
[2] Lian, B. and Kschischang, F. R., “Sequential Decoding of Short Length Binary Codes: Performance Versus Complexity,” IEEE Communications Letters, 25(10), 3195-3198(2021).
[3] Johannesson, R. and Zigangirov, K. S., [Fundamentals of Convolutional Coding], New York: Digital and Mobile Communication, 269-371(2015).
[4] Smeshko, A., Ivanov, F. and Zyablov, V., “Upper and Lower Estimates of Frame Error Rate for Convolutional Codes,” 2020 International Symposium on Information Theory and Its Applications (ISITA), IEEE, 160-164(2020).
[5] Çavdar, T. and Gangal, A., “A new sequential decoding algorithm based on branch metric,” Wireless Personal Communications, 43(4), 1093-1100(2007).
[6] Wu, Y. J., Fu, G. and He, Q., “Study of VLF communication technique based on FRFT,” Applied Mechanics and Materials, 556, 5116-5120(2014).
[7] Maurya, A. K., Prakash, R., Sriwas, S. K., Kumar, P. and Singh, R. K., “Impact of code rate and constraint length variation on qualitative performance of OIDMA system with random inter-leaver,” Optik, 202, 163584 (2020).
[8] Han, Y. S., Wu, T. Y., Chen, P. N. and Varshney, P. K., “A low-complexity maximum-likelihood decoder for tail-biting convolutional codes,” IEEE Transactions on Communications, 66(5), 1859-1870(2018).
[9] Lavoie, P., and Haccoun D., “A systolic architecture for fast stack sequential decoders,” IEEE Transactions on Communications, 42(2), 324-335(1994).
[10] Jeong, M. O. and Hong, S. N., “SC-Fano decoding of polar codes,” IEEE Access 7, 81682-81690(2019).
[11] Xiang, L., Liu, Y., Maunder, R. G., Yang, L. L. and Hanzo, L., “Soft-output successive cancellation stack polar decoder,” IEEE Transactions on Vehicular Technology, 70(6), 6238-6243(2021).
[12] Kakacak, A. and Kocak, T., “Design and implementation of high throughput bidirectional Fano decoding,” 2013 8th IEEE Conference on IEEE Industrial Electronics and Applications (ICIEA), 1670-1675(2013).
[13] Asher, Y. B., Tartakovsky, V., Portman, K., Zilberman, O. and Hadar, A., “An FPGA scalable parallel viterbi decoder,” 2018 IEEE 12th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC), IEEE, 8-15(2018).
[14] Lin, S. and Costello, D. J., [Error Control Coding], New York: Prentice Hall, 2(4), 354-453(2001).

[15] Jelinek F., “A fast sequential decoding algorithm using a stack,” IBM Journal of Research and Development, 13(6), 675-685(1969).
[16] Song, W., Zhou, H., Niu, K., Zhang, Z., Li, L., You, X. and Zhang, C., “Efficient successive cancellation stack decoder for polar codes,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 27(11), 2608-2619(2019).
[17] Hou W. and Jiang Y., “Simple hardware implementation of soft decision sequential decoder,” International Conference on Communications Technology and Applications (ICCTA), 687-691(2009).
[18] Smeshko, A., Ivanov, F. and Zyablov, V., “Upper and lower estimates of frame error rate for convolutional codes,” 2020 International Symposium on Information Theory and Its Applications (ISITA), IEEE, 160-164(2020).